PCI Hot-Plug Specification

Revision 1.1

June 20, 2001

Revision	Revision History	Date
1.0	Original issue.	October 6, 1997
1.1	Add PCI 2.2, PCI power management, PCI-X, and SMBus.	June 20, 2001

PCI-SIG disclaims all warranties and liability for the use of this document and the information contained herein and assumes no responsibility for any errors that may appear in this document, nor does PCI-SIG make a commitment to update the information contained herein.

Contact the PCI-SIG office to obtain the latest revision of the specification.

Questions regarding the PCI Hot-Plug Specification or membership in PCI-SIG may be forwarded to:

Membership Services
http://www.pcisig.com
E-mail: administration@pcisig.com
Phone: 503-291-2569
Fax: 503-297-1090

Technical Support
Technical Support for this specification is available to members. For information, please visit: http://www.pcisig.com/developers/technical_support.

DISCLAIMER

This PCI Hot-PlugSpecification is provided “as is” with no warranties whatsoever, including any warranty of merchantability, noninfringement, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. PCI-SIG disclaims all liability for infringement of proprietary rights, relating to use of information in this specification. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein.

PCI-SIG is a trademark of PCI-SIG.

All other product names are trademarks, registered trademarks, or servicemarks of their respective owners.

Contents

Chapter 1 Introduction

Chapter 2 Overview of PCI Hot Plug

Chapter 3 Hardware and Firmware Requirements

Chapter 4 Software Requirements

Appendix A Presence and Capability Signals

Figures

Tables

Chapter 1
Introduction

1.1 Objectives of the PCI Hot-Plug Specification

The primary objective of this specification is to enable higher availability of file and application servers by standardizing key aspects of the process of removing and installing PCI add-in cards while the system is running. Although these same principles can be applied to desktop and portable systems using PCI buses, the operations described here target server platforms.

The burden of hardware changes has been placed on the hardware platform vendor, not the add-in card vendor. This specification describes a new class of hardware platforms, *not* a new class of add-in cards. Electrical requirements for the add-in card, beyond those already stated in the *PCI Local Bus Specification*, have been kept to an absolute minimum.

It is the expressed intent of this document to do the following:

- Specify what an add-in card vendor must do to enable an add-in card to be hot-pluggable.
- Specify required features of the hot-plug hardware platform, so operating-system vendors can be assured of a minimum feature set. The list of required platform features has been kept short to enable the implementation of cost-sensitive PCI hot-plug server hardware platforms.
- Establish a framework of terminology and architecture for hot-plug system hardware and software.
- Assign responsibility for each component and operation either to the hardware platform vendor, the operating-system vendor, or the add-in card vendor for the general application of this specification. However, this specification does not preclude the delivery of components by other means that are mutually agreeable to the vendors involved.

It is expressly *not* the intent of this document to do the following:

- Specify the detailed implementation of the PCI hardware platform. To the greatest extent possible, the design of the PCI hardware platform has been left flexible to allow for product differentiation.
- Specify the implementation or structure of hot-plug software routines. Concepts and features are presented, but implementation is strictly determined by each operating-system vendor.
- Specify higher-level operating-system behavior, such as what should happen to a file system while its disk controller card is being replaced. Such functions are allowed to vary from one operating system to another and are controlled by each operating-system vendor.
- Specify higher-level hot-plug operations, such as like-for-like replacement of cards versus adding a new card. The hardware and low-level software components described here are required by all hot-plug systems and result in various implementations at the discretion of the operating-system vendor.

1.2 Areas of Standardization

Figure 1-1 illustrates some of the hardware and software components of a hot-plug system. In the figure, the *Conventional System Software* box represents applications, system management functions, the operating system, and device drivers for peripherals, including PCI add-in cards, which are present in conventional, non-hot-plug systems. The *Conventional Platform Hardware* box includes the CPU(s) and peripheral devices. The cloud in the center represents software and hardware required to control power and bus connections on PCI slots that accept standard PCI add-in cards.

This specification allows the software and hardware *within* the cloud to change and standardizes the two *interfaces* of the cloud shown in Figure 1-1. The first standard interface is the hardware interface between the platform and the add-in card. In most respects, this interface is a standard PCI interface. This document describes additional aspects of powering up and down a card plugged into a PCI connector that are beyond the scope of the *PCI Local Bus Specification*.

The second standard interface shown in Figure 1-1 is the software interface between the higher-level system software and the platform-specific hot-plug software. The information *content* of the requests and responses that cross this interface are specified in this document. However, since this is a device-driver interface, the information *format* varies from one operating system to another and is controlled by each operating-system vendor.

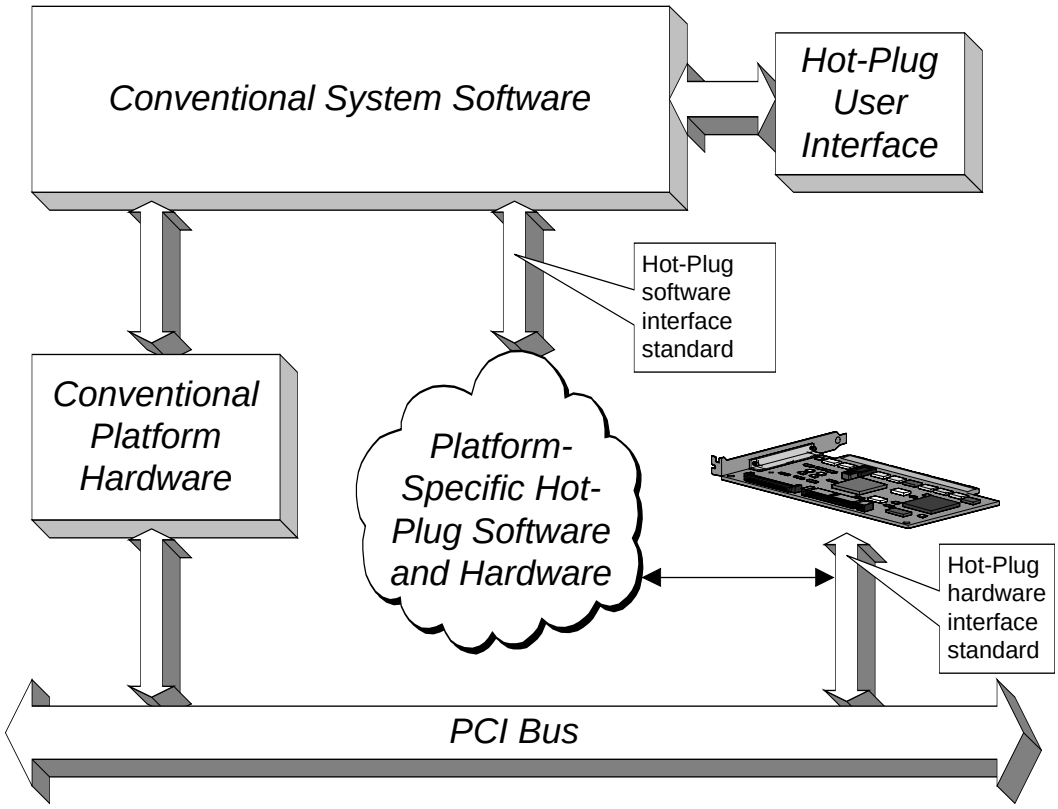

Figure 1-1: Standardized Hot-Plug Software and Hardware Interfaces

1.3 Reference Documents

The following documents are cited in this specification:

PCI Local Bus Specification, Revision 2.2 (PCI 2.2), December 18, 1998, PCI Special Interest Group

PCI-X Addendum, Revision 1.0a (PCI-X 1.0a), July 24, 2000, PCI Special Interest Group

PCI Bus Power Management Interface Specification, Revision 1.1 (PCI PM 1.1), December 18, 1998, PCI Special Interest Group

Addition of SMBus Interface to Connector (SMBus ECN), August 9, 2000, PCI Special Interest Group

PCI Standard Hot-Plug Controller and Subsystem Specification, Revision 1.0 (SHPC 1.0), June 20, 2001, PCI Special Interest Group

1.4 Notational Conventions

As in PCI 2.2, references in this specification to 33-MHz operation refer to a bus that operates with a clock frequency from 0 to 33 1/3 MHz, and references to 66-MHz operation (in conventional PCI mode) refer to a bus that operates with a clock frequency greater than 33 1/3 MHz up to 66 2/3 MHz. As in PCI-X 1.0a, references to 66-MHz operation (in PCI-X mode) refer to a bus that operates with a clock frequency greater than 50 MHz up to 66 2/3 MHz, references to 100-MHz operation refer to a bus that operates with a clock frequency greater than 66 2/3 MHz up to 100 MHz, and references

to 133-MHz operation refer to a bus that operates with a clock frequency greater than 100 MHz up to 133 1/3 MHz.

Signal names are indicated with this **bold** font. Collections of signals that are collectively driven and received are assigned the same signal name with numbers in brackets to indicate the bit or bits affected, for example, **PRSNT[1::2]#**.

1.5 Definitions of Terms

Definitions of the terms used in this specification are listed below. Most terms are capitalized throughout the remainder of this specification to call attention to their special definitions.

add-in card

A card designed in accordance with PCI 2.2 to be plugged into a PCI connector. This includes cards that are 32- or 64-bits wide, operate in conventional or PCI-X mode at 33 MHz, 66 MHz, or 133 MHz, and use 3.3 V or 5 V signaling. The card is permitted to contain a single PCI device or multiple devices behind a PCI-to-PCI bridge.

Since most of the additional hardware requirements of a PCI hot-plug system are provided by the platform, most add-in cards designed to earlier versions of the *PCI Local Bus Specification* are also allowed. However, no attempt is made in this specification to identify issues with add-in cards not compliant with PCI 2.2.

add-in card driver

The software driver which controls the add-in card. It is generally supplied by the add-in card vendor.

Attention Indicator

A physical indicator, located to draw the attention of the user to a particular slot. The Platform is required to provide one Attention Indicator per hot-plug slot. The state of the Attention Indicator is determined by the Hot-Plug Service.

Hot-Plug Controller

Hardware supplied by the Platform vendor that controls the electrical aspects of powering up and down a PCI slot. A single Hot-Plug Controller typically controls more than one slot. A hot-plug Platform is permitted to contain more than one Hot-Plug Controller.

Hot-Plug Primitives

Specific requests issued by the Hot-Plug Service to the Hot-Plug System Driver to determine the status of, or to initiate changes to, a hot-plug slot in the Platform.

Hot-Plug Service

High-level software that has overall control of hot-plug operations. It includes a user interface, and issues requests to the operating system to quiesce add-in card activity, and issues Hot-Plug Primitives to the Hot-Plug System Driver to turn slots on and off. The Hot-Plug Service is unique to each particular operating system and is generally supplied by the operating-system vendor.

hot-plug slot

A slot designed in accordance with this specification to allow the insertion and removal of add-in cards without powering down the Platform or restarting the operating system. This is the basic unit of hot-plugability. Individual slots must be isolated from the rest of the Platform for reliable insertion and removal of an add-in card.

Hot-Plug System Driver

Software driver that controls and monitors the Hot-Plug Controller hardware. If there is more than one Hot-Plug Controller in a Platform, some operating systems require more than one Hot-Plug System Driver. The Hot-Plug System Driver is supplied by the Platform vendor.

Logical Slot Identifier

A parameter of a Hot-Plug Primitive that uniquely identifies a particular slot. The operating-system vendor specifies the encoding of this parameter. For example, some vendors use PCI bus and device number, while other vendors use physical slot numbers.

Physical Slot Identifier

A designation assigned by the Platform vendor that uniquely identifies a physical slot. For example, in a single-chassis system, it is commonly a slot number. In some multiple-chassis systems, it is a combination of chassis and slot number.

Platform

The collection of hardware in which the PCI bus resides. Generally includes the power supply, one or more CPUs, a host-bus-to-PCI bridge, and various peripherals such as disk drives, a keyboard, and so on.

Platform Configuration Routine

Software responsible for initializing the PCI Configuration Space header for a newly installed add-in card. The operating-system vendor specifies whether this routine is executed by the Hot-Plug Service or by the Hot-Plug System Driver.

quiesce Before an add-in card in a slot can be removed, card activity must be quiesced. When card activity is quiesced, the add-in card driver does not send any PCI operations to the card, and the card does not initiate any interrupts or bus master activity. See Section 4.1.2 for more details.

slot A location designed to accept an add-in card.

1.6 Assumptions

1.6.1 PCI Add-in Card Failure

The PCI bus inherently is not a fault-tolerant bus. A failure of a PCI add-in card can have a wide range of effects on system behavior. For example, the failure might be as harsh as preventing further PCI bus transactions or corrupting the contents of main memory. Or the failure might be as localized as losing a single network connection. If a failure of a device compromises the integrity of the PCI bus or the software system, the only recourse is to restart the system. It is assumed throughout the remainder of this specification that device failures that compromise the integrity of the system are beyond the scope of this specification. Only failures that do not compromise the integrity of the system are relevant to this specification.

1.6.2 Orderly Removal and Insertion

The operating systems that are widely used throughout the PCI industry are not generally designed to handle the unexpected removal of devices. Therefore, the operating-system vendor and Platform vendor define the sequence of user actions and system management facilities that inform the operating system that removal of a card is desired. The actual removal cannot occur until the operating system acknowledges that it is ready.

Furthermore, PCI add-in cards are not generally designed to be connected to a slot that has power applied. Therefore, the operating-system vendor and Platform vendor define a sequence of user actions and system behavior that guarantees that power is always removed from a slot before a card is inserted.

Inserting or removing an add-in card without following the proper sequence leads to unpredictable results, including data corruption, abnormal termination of the operating system, or damage to card or Platform hardware. Unless otherwise stated, it is assumed throughout the remainder of this specification that the user always follows the proper removal and insertion sequence.

1.6.3 Programmatic Access to the Hot-Plug Service

This specification describes the required function of high-level software, but not its form or structure. The operating-system vendor exclusively controls the form and structure. However, operating-system vendors commonly provide programmatic interfaces to hot-plug functionality. These interfaces enable additional software to add other functions such as:

- Remote power down of intermittent or bad PCI cards
- Diagnostic packages
- Remote/Local tracking/display of system resources and status
- Remote/Local notification of configuration changes
- Remote/Local control of hot-plug operations

It is assumed throughout the remainder of this specification, whenever a user interface is described, that the operating system provides an equivalent programmatic interface. See Section 4.1.7, for more details on the user interface.

1.7 Standard Hot-Plug Controller

SHPC 1.0 specifies an implementation of a Hot-Plug Controller that is compliant to this specification. By standardizing the Hot-Plug Controller, SHPC 1.0 makes it possible for a single Hot-Plug System Driver to be supplied by the operating-system vendor that supports multiple hot-plug Platforms. Although the remainder of this specification assumes the Hot-Plug System Driver is supplied by the Platform vendor, hot-plug Platforms that are compliant with SHPC 1.0 generally use the Hot-Plug System Driver supplied by the operating-system vendor.

Chapter 2
Overview of PCI Hot Plug

2.1 System Components

Figure 2-1 illustrates the hardware and software components of a typical hot-plug system and how they are connected.

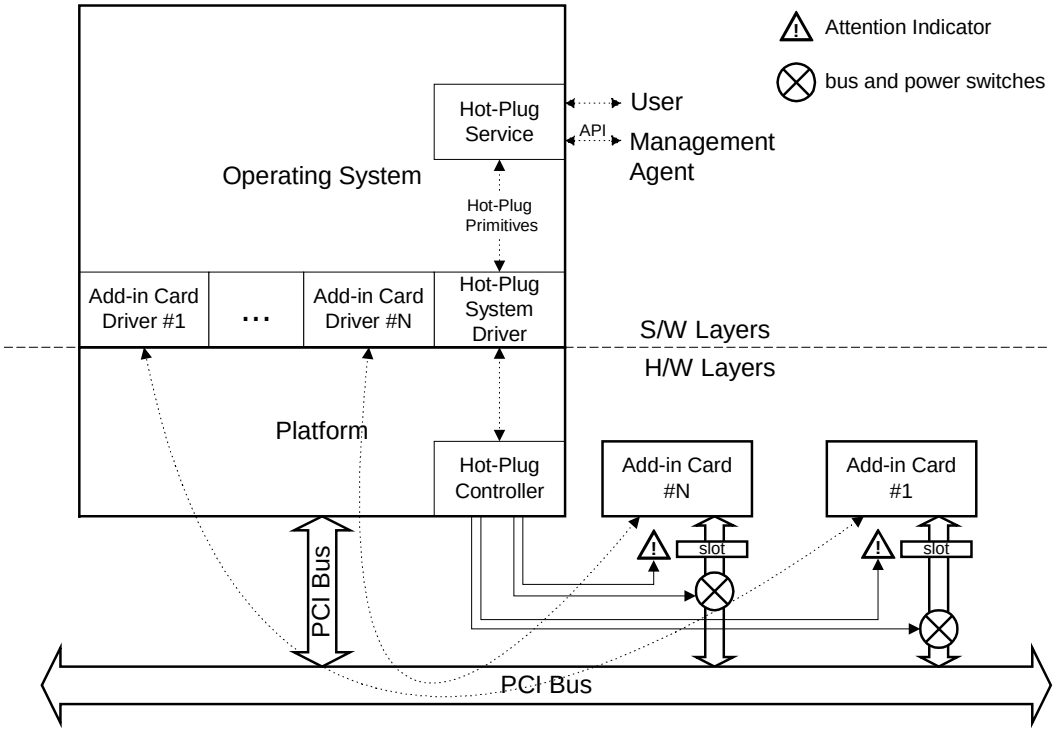

Figure 2-1: System Block Diagram

The electrical capabilities that the Platform vendor must provide are defined in Chapter 3. They are summarized as follows:

- The ability to isolate an individual slot from its associated PCI bus
- Individual control of the power and control signals for a slot such that the slot is isolated and powered down during add-in card insertion or removal
- Power-up and power-down sequences for hot-plug slots that meet the electrical requirements of PCI 2.2 and PCI-X 1.0a (if the slot is capable of PCI-X operation)
- An Attention Indicator to draw the user's attention to a particular slot

The Platform and system software (operating system and drivers) must define the sequence of user actions that constitutes a proper hot-insertion or removal sequence. This proper sequence is the actions that a user is required to perform to inform the operating system that an insertion or removal of an add-in card is desired. The physical insertion or removal must not occur until the system software has:

- Quiesced any operating system services or drivers using the add-in card
- Isolated and powered down the slot
- Indicated to the user that it is ready

If an add-in card is inserted or removed without following the proper sequence, this is considered an improper operation and error conditions and other unexpected events are possible, including data corruption and hardware damage.

2.2 Illustration of a Hot-Plug Sequence

The following hot-plug scenarios for removing and inserting an add-in card are given for example and overview purposes and do not represent requirements upon either Platforms or operating systems. For any given system, the Hot-Plug Service, the Hot-Plug System Driver, and the Hot-Plug Controller each control different portions of the actual order.

2.2.1 Hot Removal

The following general sequence of steps is necessary to remove an add-in card from a slot that is powered up:

1. The user determines that an add-in card must be removed or replaced and notifies the Hot-Plug Service of the desire to remove the card from its slot. Examples of notification methods include issuing a console command or activating a switch designed for this purpose.
2. The Hot-Plug Service uses operating system functions to quiesce the appropriate add-in card driver instance(s) and the add-in card.
3. The Hot-Plug Service issues a Hot-Plug Primitive to the Hot-Plug System Driver to turn off the appropriate slot.

4. The Hot-Plug System Driver uses the Hot-Plug Controller to do the following:
   a) Assert **RST#** to the slot and isolate the slot from the rest of the bus, in either order.
   b) Power down the slot.
   c) Change the optional slot-state indicator, as defined in Section 3.1.1, to show that the slot is off.
5. The Hot-Plug Service reports to the user that the slot is off.
6. The user removes the add-in card.

2.2.2 Hot Insertion

A slot must be powered down and isolated from the bus before an add-in card can be inserted. The process of making a slot ready for insertion depends upon the particular Platform and operating system. The following general sequence of steps is necessary to insert an add-in card into a slot after it is powered down and ready for insertion:

1. The user inserts the new add-in card.
2. The user notifies the Hot-Plug Service to turn on the slot containing the new add-in card.
3. The Hot-Plug Service issues a Hot-Plug Primitive to the Hot-Plug System Driver to turn on the appropriate slot.
4. The Hot-Plug System Driver uses the Hot-Plug Controller to do the following:
   a) Power up the slot.
   b) Deassert **RST#** on the slot and connect the slot to the rest of the bus, in either order.
   c) Change the optional slot-state indicator, as defined in Section 3.1.1, to show that the slot is on.
5. The Hot-Plug Service notifies the operating system that the new add-in card is installed, and the operating system initializes the add-in card and prepares to use it.
6. The Hot-Plug Service notifies the user that the card is ready.

Chapter 3
Hardware and Firmware Requirements

3.1 Platform Requirements

3.1.1 Supplementary Hardware Requirements

A hot-plug Platform must provide the following features beyond what is specified in PCI 2.2, PCI-X 1.0a, and PCI PM 1.1:

1. At least one Hot-Plug Controller.
2. Slot-specific power switches. The Platform must provide a means for the software to remove all main power (that is, all power other than **3.3Vaux**) from a slot while the rest of the Platform is running.

   As described in PCI PM 1.1, a Platform optionally provides **3.3Vaux** to a slot to power logic on the card that needs to remain active when the rest of the system is unpowered (for example, modem ring indicator detection logic). A Platform with hot-plug slots must guarantee that **3.3Vaux** is removed from such a slot prior to inserting or removing a card, even if the insertion and removal occurs while system power is off and no software is running.
3. Slot-specific bus isolation devices. The Platform must provide a means for the software to isolate all the signals on a slot (except **PME#** and SMBus) from the bus, while the rest of the Platform is running.

   If the Platform supports **PME#** connections to the slots (as defined in PCI PM 1.1) or SMBus connections to the slots (as defined in the SMBus ECN), the Platform must guarantee that these signals are isolated from the bus prior to inserting or removing a card, even if the insertion and removal occurs while system power is off and no software is running.

   In all cases, the Platform is required to meet all the AC timing specifications of PCI 2.2, PCI-X 1.0a, PCI PM 1.1, and the SMBus ECN at the slot when that slot's isolation devices are in the conducting mode. The Platform must also guarantee that

all of the slot's signal pins are biased to a valid logic level, as defined in PCI 2.2, when the slot's isolation devices are in the isolating mode and power is applied to the slot.

4. Slot-specific **RST#**. During hot plug events, the Platform must provide a means to control the **RST#** pin to the affected slot and all signals that are required by PCI 2.2 or PCI-X 1.0a to be set up to the rising edge of **RST#**. Signals affected include **REQ64#**, and (for PCI-X implementations) **DEVSEL#**, **STOP#**, and **TRDY#**. (For slots that are 32 bits wide and operate only in conventional mode, the Platform deasserts all these signals at the rising edge of **RST#**.) Refer to the appropriate PCI specifications and addendums for detailed signaling requirements. Refer to Section 3.1.3 for additional information.
5. Slot-specific Attention Indicator. Each hot-plug slot must have an Attention Indicator that is located to draw the attention of the user to that slot, for example, an LED located near the slot. The state of the Attention Indicator is controlled by the software.
6. A means for software to determine the state of the **PRSNT[1::2]#** and (for PCI-X implementations) **PCIXCAP** pins for each hot-plug slot regardless of whether the slot is on or off and connected or isolated.
7. A means for software to determine the current bus mode and clock frequency, as defined in PCI 2.2 and PCI-X 1.0a.
8. A means for software to determine the state of the **M66EN** pin for each 66-MHz hot-plug slot, while the slot is isolated from the bus. A Platform is not required to implement this means if either of the following is true:

   a) The Platform hardware guarantees that a 33-MHz card is not connected to the bus if the bus is operating at 66 MHz.

   b) The bus includes no devices other than the source bridge and a single slot, and the Platform hardware or the Hot-Plug System Driver guarantees that the bus is operating at 33 MHz before connecting a card.

   See Section 3.4 and Appendix A for additional information relating to **M66EN**.
9. A Physical Slot Identifier associated with each hot-plug slot.
10. Information accessible by the software that the operating-system vendor uses to define translation mechanisms between Physical Slot Identifier and the PCI bus and device number associated with that slot (see Section 4.1.8).
11. A Platform that supports SMBus connections to a slot (as defined in SMBus ECN) must provide a means for the software to supply main power to the slot without connecting the slot to the bus.

Implementation Note: PME# and SMBus Connection

As stated in Item 3 above, if a Platform supports **PME#** or SMBus connections to the slots, that Platform must provide a means to isolate them from the bus.

On some cards, **PME#** or SMBus devices are powered by **3.3Vaux** and are used while the system is in a low-power state. For example, in some Platforms, the slot's **PME#** signal remains connected to the bus while the system is in a low-power state, to allow the card to wake the system. One method of supporting such cards is for the Platform to connect **PME#** and the SMBus to the slot whenever **3.3Vaux** is connected. In such an implementation, the same user action that causes the connection of **3.3Vaux** would also cause the connection of **PME#** and the SMBus, and the same user action that causes the removal of **3.3Vaux** would also cause the isolation of **PME#** and the SMBus, even while system power is off.

On other cards, the SMBus devices are powered by main power. To support these cards, the Platform provides a means for software to apply power to the slot before accessing the SMBus device, as stated in Item 11 above. Providing such a means enables the software to access SMBus devices on the card, even if the card cannot be connected to the bus because the bus is running in a mode or clock frequency that is not supported by the card (see Section 3.4). In many Platforms, the same mechanism that is used to apply power to the slot to determine the state of **M66EN** (see Appendix A) is also employed to enable software access to SMBus devices before connecting the card to the bus.

Note that if the SMBus is connected to the slot before the rest of the bus, the only clock the SMBus device receives is the SMBus clock, **SMBCLK**. SMBus requirements always assume that only **SMBCLK** is necessary. However, if an SMBus device also uses the PCI clock, the software is unable to access that device until the slot is fully connected to the bus.

A hot-plug Platform is permitted to provide the following optional features:

- Slot-specific slot state indicator. Each hot-plug slot may have an indicator that shows whether the slot is on or off. If implemented, this indicator must reflect the current slot state as controlled by the Setting Slot Status Hot-Plug Primitive. See Section 4.2.2.2.

  An implementation that combines the slot state indicator with the Attention Indictor into a single device is permitted. In such implementations, the attention state is required to take precedence over the slot state, if both states cannot be shown simultaneously. For example, a single LED might be implemented that is on if the slot is on, off if the slot is off, and blinking in the attention state whether the slot is on or off.

- Slot-specific power fault detector. The Platform may include the capability of detecting when an add-in card exceeds the power limits of a slot (see Section 3.3) and automatically turning off the slot.

- System power budget. The Platform vendor may implement a means of tracking power usage and determining whether there is sufficient power available to turn on an additional add-in card.

The Platform vendor is permitted to implement other features but is responsible for providing all necessary software support for them.

3.1.2 Platform Initialization

If no hot-plug software (other than Platform firmware) is loaded onto a hot-plug Platform, and no hot-insertion or removal operations are performed, the Platform must behave as if it were not a hot-plug Platform. After power is initially applied to a hot-plug system at the time when the operating system begins loading, Platform firmware must ensure that all Attention Indicators are off, and the hot-plug slots are in a state that would be appropriate for loading non-hot-plug system software.

3.1.3 Turning On a Slot

If a slot is available for an add-in card to be inserted while the rest of the Platform is running, the power is removed from the connector, and the connector is isolated from the PCI bus. The Platform vendor must guarantee that, when turning on a slot, the slot behaves in full accordance with PCI 2.2 and PCI-X 1.0a.

The first step in turning on a slot is turning on the power. The steady-state supply load and amount of decoupling capacitance on the add-in card are specified in Section 3.3. The Platform is required to turn on power to a slot containing an add-in card (that meets these load requirements) in a way that does not interfere with the operation of the rest of the Platform and that meets all the requirements of PCI 2.2. Some voltage ramp characteristics are specified in Section 3.3. The Platform vendor determines all other voltage and current ramp characteristics.

After the supply voltages are stable, the Hot-Plug Controller must electrically connect the signals on the slot to the bus and deassert **RST#**. The Platform is permitted to connect the slot to the bus either before or after **RST#** is deasserted, but, in either case, must meet all the setup- and hold-time requirements of PCI 2.2 and PCI-X 1.0a with respect to the rising edge of **RST#**.

Transactions between other devices occur before and after the slot is connected and **RST#** is deasserted. The Platform is permitted to control the PCI bus at the time the bus is connected and the time **RST#** is deasserted to prevent interference with other transactions.

Implementation Note: REQ64# and the PCI-X Initialization Pattern

PCI 2.2 and PCI-X 1.0a define how **REQ64#** and the PCI-X initialization pattern (driven on **DEVSEL#**, **STOP#**, and **TRDY#**) indicate the width and operating mode of the bus at the rising edge of **RST#**. This specification requires the Platform to provide slot-specific **RST#** signals so that individual slots can be enabled after other devices are already operating. The Hot-Plug Controller in a system that implements a 64-bit bus or that operates in PCI-X mode must provide a means to apply the appropriate values to these signals at the rising edge of slot-specific **RST#**. (For slots that are 32 bits wide and operate only in conventional mode, the Platform deasserts all these signals at the rising edge of **RST#**.)

This specification allows the deassertion of **RST#** and connection of the bus in either order. However, in 64-bit and PCI-X systems, it is generally preferable to connect the slot to the bus before deasserting **RST#**. If the bus is connected first, a single central resource is able to drive **REQ64#** and the PCI-X initialization pattern, regardless of which slot or slots are being connected. However, if the bus is connected to the slot after **RST#** is deasserted, special steps (which are not described in this specification) must be taken to steer these signals on the slot side of the bus isolation devices at the rising edge of **RST#**.

3.1.4 Turning Off a Slot

When a slot is turned off, the Platform is required to assert **RST#** to the slot and electrically isolate the slot from the bus, in either order, and then to remove power from the slot. Power-down timing must comply with PCI 2.2.

3.2 Add-in-Card Requirements

3.2.1 Noteworthy Requirements of the PCI Local Bus Specification and the PCI-X Addendum

The following requirements of PCI 2.2 and PCI-X 1.0a are of particular interest during hot-plug operations with an add-in card, since violations could potentially go unnoticed in a non-hot-plug system:

1. **PRSNT[1::2]# connection.**[1] One or both of these pins must be grounded by add-in cards to indicate that a card is present in the slot and how much power the card requires. **PRSNT[1::2]#** pins not grounded must be left unconnected. Hot-plug Platforms read these pins to determine which slots are occupied and how much power the card requires.
2. **State of outputs when RST# asserted.**[2] All bus outputs from the add-in card must be placed in their benign state asynchronously when **RST#** is asserted. Tri-state and open drain signals must float, and totem pole outputs must deassert. The assertion of

[1] PCI 2.2, Sections 2.2.7, 4.3.4.1, and 4.4.1.

[2] PCI 2.2, Sections 2.2.1 and 4.3.2.

RST# must cause **INTA#**, **INTB#**, **INTC#**, and **INTD#** to deassert. In hot-plug Platforms, the bus is potentially running transactions between other active devices while a device being hot-plugged has its **RST#** pin asserted.

3. **PCI interface state machines' response to RST#.**[3] All PCI interface state machines must be held in their reset state as long as **RST#** is asserted. In hot-plug Platforms, a slot with **RST#** asserted is potentially connected to an active bus carrying transactions between other devices. These transactions addressing other devices must have no effect on the PCI interface of the device being reset.

   Once **RST#** is deasserted, the PCI interface state machines remain in the IDLE state until an address phase containing a valid address for the device is encountered.

Implementation Note: Slow Recognition of Deassertion of RST#

Some PCI devices may require an extended period of time to initialize the PCI interface after **RST#** is deasserted. For example, the device may need to wait for a local PLL to lock, or for configuration registers to initialize from a slow external device. If the initialization period takes more than a few clocks, such a device could come out of reset in the middle of a transaction between two other devices. (This would be true when the whole system was powered up as well as when a card was hot-plugged.) To avoid misinterpreting a transaction that is already in progress, such a device must not leave the reset state unless the PCI bus is idle.

4. **Behavior of CLK prior to the rising edge of RST#.**[4] PCI 2.2 requires **CLK** to be stable a certain length of time before the rising edge of **RST#**. There are no specifications for **CLK** prior to this time. The add-in card must not depend on **CLK** to have any particular timing characteristics or valid logic levels prior to its setup to the rising edge of **RST#**.

Implementation Note: Examples of CLK Behavior Prior to Rising Edge of RST#

Some examples of how **CLK** might behave prior to its stable setup to **RST#** are:

- **CLK** could remain at a logic low level continuously from the time power is first applied until the bus signals are connected, then change instantaneously to the proper frequency.
- While **CLK** is being connected, pulse widths might be of any value, and logic levels may not be valid.
- **CLK** could begin toggling between ground and the ramping supply voltage, while the supply voltage is ramping.
- **CLK** could oscillate at various frequencies for various periods of time or continuously vary in frequency.

Other examples are also possible.

[3] PCI 2.2, Section 2.2.1 and Appendix B.

[4] PCI 2.2, Section 4.2.3.2.

5. **Preference for registers to be mapped into Memory Space rather than I/O Space.**[5] PCI 2.2 highly recommends the use of Memory Space over I/O Space, because I/O Space is limited and fragmented in PC Platforms. The limitation and fragmentation of I/O Space will be even more severe in hot-plug Platforms after the system has booted.

6. **Efficient resource requests.**[6] Although PCI 2.2 permits a device to consume more address space than it needs, it suggests decoding down to a 4 KB space for devices that require less than that amount of memory. Devices requiring I/O Space are required to use no more than 256 bytes per Base Address register and are required to provide memory mapping for the registers as well.

   Increasing the amount of resources a device requests will decrease the likelihood that the Platform Configuration Routine can configure new add-in cards added after the system boots. Decoding to the smallest address space possible (even smaller than 4 KB of memory and 256 bytes of I/O space) improves the likelihood that more devices can be added without rebooting the system. Wasting resources (requesting more resources than are actually required) makes it more difficult for other add-in cards to be configured. Devices should only request resources that will actually be utilized.

7. **Switching bus frequency.**[7] If the clock frequency is above 33 MHz, PCI 2.2 and PCI-X 1.0a require a frequency change to occur in conjunction with a PCI reset. The add-in card cannot assume that such a bus frequency change will also be in conjunction with a power cycle. See also Section 3.4.

8. **Ignoring REQ64# while the bus is idle.**[8] **REQ64#** has meaning only at the rising edge of **RST#** and when **FRAME#** is asserted. The bus interface state machines must ignore the assertion of **REQ64#** at all other times (for example, while the bus is idle). Some Platforms assert **REQ64#** (with **FRAME#** deasserted) to the entire bus when the slot-specific **RST#** for a newly inserted card is deasserted. Other cards connected to the same bus segment must ignore this event.

9. **Subsystem Vendor ID and Subsystem ID valid after RST#.**[9] PCI 2.2 requires these registers in the Configuration Space header of all devices on an add-in card to be valid prior to the system BIOS or any system software accessing the device.

10. **PCIXCAP connection.**[10] An add-in card capable of PCI-X operation must connect this pin to ground through passive components. The pin must not be connected to any supply voltages or active components. Hot-plug Platforms read the state of this pin without applying power to the slot.

11. **Ignoring target response signals while the bus is idle.**[11] A device's bus interface state machines must ignore the assertion of any combination of **DEVSEL#**, **STOP#**,

[5] PCI 2.2, Section 3.2.2.

[6] PCI 2.2, Section 6.2.5.1.

[7] PCI 2.2, Section 7.6.4.1, Table 7-3, Note 1 and PCI-X 1.0a, Section 9.4.1, Table 9-4, Note 1.

[8] PCI 2.2, Section 2.2.8.

[9] PCI 2.2, Section 6.2.4.

[10] PCI-X 1.0a, Section 9.10.

[11] PCI-X 1.0a, Section 6.2.1.

and **TRDY#** while **FRAME#** and **IRDY#** are deasserted (that is, the bus is idle). Platforms capable of PCI-X operation use these signals to place newly inserted cards in PCI-X mode while other cards observe an idle bus.

3.2.2 Remote Power Sources

An add-in card is permitted to have a connection to another device that receives power from a source other than the Platform slot. The add-in card vendor must guarantee that the card's PCI interface is properly reset when **RST#** is asserted, whether the remote device is powered or not.

The add-in card vendor must further guarantee that the user is not exposed to any hazard, and that neither the add-in card nor the remote device sustains any damage if the add-in card or the remote device is powered for an indefinite length of time while the other is not.

3.2.3 Multiple-Card Sets

A multiple-card set is any group of add-in cards that is normally installed and removed together. Multiple-card sets are often interconnected with sideband cables. Some multiple-card sets appear in PCI Configuration Space as multiple PCI devices, and others appear as a single PCI device (using the additional slots only for physical space or power-consumption reasons).

Multiple-card sets that do not require that power be applied and removed to the whole set simultaneously are accommodated by all hot-plug Platforms by using multiple single-card operations. Therefore, it is recommended that all multiple-card sets be designed such that they are not adversely affected if only a portion of the set is powered for an indefinite period of time. Furthermore, each card in the set must properly connect the **PRSNT[1::2]#** pins to guarantee that power is applied to each of the slots (see Section 3.2.1).

This specification neither specifically allows nor disallows multiple-card sets that require simultaneous application and removal of power. Nor does this specification include a means for the software to uniquely identify which cards are in a multiple-card set. Although various places in the specification (for example, the Hot-Plug Primitives) assume only single-slot operation, Platform vendors are permitted to accommodate such multiple-card sets and develop all the necessary hardware and software to support them.

3.3 Slot Power Requirements

Table 3-1 describes power requirements to ensure that an add-in card can be inserted into a running system.

The first requirement is the maximum operating current drawn by an add-in card, defined as current drawn by all loads other than the decoupling capacitors. These values are specified in PCI 2.2 and are repeated in Table 3-1 for clarity. Add-in cards that have a range of operating currents, or that have a dynamic or switching load, must guarantee that the peak operating loads never exceed the values shown in Table 3-1, even while the add-in card's power is turning on. For example, a switching voltage regulator must maintain the peak operating current below the specified maximum even when initially charging the secondary decoupling capacitors.

The second requirement is the maximum decoupling capacitance on an add-in card. Add-in card decoupling capacitance must not exceed the values shown in Table 3-1.

The third and fourth requirements involve power supply voltage characteristics when a slot is being turned on. The Platform must ensure that the supply voltage slew rate during power up is between the minimum and maximum values in Table 3-1. The add-in card must tolerate supply voltage slew rates between this minimum and maximum without suffering damage.

Table3-1: Slot Power Requirements

Supply Voltage	Maximum Operating Current (Note 1, 2)	Maximum Add-in Card Decoupling Capacitance	Minimum Supply Voltage Slew Rate	Maximum Supply Voltage Slew Rate
+5 V	5 A	3000 µF	25 V/s	3300 V/s
+3.3 V	7.6 A	3000 µF	16.5 V/s	3300 V/s
+12 V	500 mA	300 µF	60 V/s	33000 V/s
-12 V	100 mA	150 µF	60 V/s	66000 V/s
3.3 Vaux	375 mA	150 µF	16.5 V/s	3300 V/s

Notes:

1. This parameter is specified by PCI 2.2[12] or PCI PM 1.1[13] and is included here for reference purposes only.
2. As specified in PCI 2.2, combined maximum power drawn by all supply voltages in any one slot must not exceed 25 W.[14]

Implementation Note: Charging Decoupling Capacitance

When powering up a slot, the Platform must provide not only the operating current for the add-in card (up to the maximum specified in Table 3-1) but also current to charge the decoupling capacitance on the card. The Platform controls the in-rush current to the card by limiting the slew rates of the supply voltages (within the limits specified in Table 3-1). For example, if a card had 3000 µF of decoupling capacitance on the +5 V supply, and the Platform turned on the +5 V supply at 3300 V/s, the in-rush current would be up to 5 A for the operating current plus 9.9 A for charging the decoupling capacitance. Limiting the +5 V supply turn-on rate to 25 V/s would limit the in-rush current to 5 A for the operating current plus 75 mA for charging the decoupling capacitance.

[12] PCI 2.2, Section 4.3.4.1.

[13] PCI PM 1.1, Section 5.4.3.

[14] PCI 2.2, Section 4.4.2.2.

3.4 Bus Mode and Frequency Considerations

PCI 2.2 and PCI-X 1.0a require that buses not exceed the mode or frequency capability of any device on the bus.[15] The Hot-Plug System Driver is required to guarantee that a slot is not connected to the bus if the slot contains an add-in card that is not capable of operating at the present mode and frequency of the bus. See Section 4.2.2.2 for more information.

The Platform vendor is solely responsible for determining when and how the bus operating mode and frequency are switched. However, PCI 2.2 and PCI-X 1.0a require that, for clock frequencies above 33 MHz, the bus mode and frequency change only in conjunction with a PCI reset.[16] Therefore, the Platform must not switch the bus operating mode or frequency (above 33 MHz) and the Platform must not change the **M66EN** pin for a particular slot, if the following conditions are all true:

- An add-in card is present in the slot.
- The slot's power is on.
- The slot's **RST#** pin is deasserted.

[15] PCI 2.2, Section 7.5.1 and PCI-X 1.0a, Section 6.1.2.

[16] PCI 2.2, Section 7.6.4.1, Table 7-3, Note 1 and PCI-X 1.0a, Section 9.4.1, Table 9-4, Note 1.

Chapter 4
Software Requirements

This chapter presents requirements for two categories of software that support inserting and removing add-in cards while the system is running.

The first category is system software, which consists of the operating system, add-in card drivers, and the Hot-Plug Service. With a few exceptions, it is not the purpose of this specification to divide system software functionality between these three sections. Rather, this specification discusses requirements that the operating system must guarantee, either by itself or by assigning duties to the Hot-Plug Service or add-in card driver. The operating-system vendor determines the interfaces between these three sections.

The second category is Platform-specific software, which consists of the Hot-Plug System Driver and its programming interface, the Hot-Plug Primitives.

4.1 System Software

4.1.1 Items Specified by the Operating-System Vendor

The following is a summary of architectural issues or alternatives that must be specified by an operating-system vendor to enable the development of hot-plug add-in card drivers and Hot-Plug System Drivers.

Add-in Card Driver Specification:

- How the add-in card driver is notified to quiesce card activity.
- Whether the operating system supports “pausing” an add-in card driver in expectation that a similar card will be reinstalled. If so, how the operation is initiated and how additional error conditions are handled.
- What responsibility the add-in card driver has for replacing the functionality of option ROMs on the add-in card.

- How the add-in card driver is notified to start using a new add-in card or resume using an old one.

Hot-Plug System Driver Specification:

- Format and structure of the Hot-Plug Primitives and their parameters, including the Logical Slot Identifier parameter.
- Whether the Hot-Plug System Driver is responsible for executing the Platform Configuration Routine after a hot-insertion. If so, whether partial initialization of the Configuration Space header is acceptable and the state of the slot after a failure during Configuration Space header initialization.

The operating-system vendor also specifies the Hot-Plug Service and its user interface and/or programming interface for remote management. See Section 4.1.7 for more details.

4.1.2 Quiescing Add-in Card Activity

Before an add-in card can be removed from a Platform, the system must stop accessing the card, and the card must stop accessing the system. The sequence of steps that the system uses to make this guarantee is called quiescing add-in card activity. The operating-system vendor must define the method of quiescing and restarting add-in card activity. Although the details of this operation are strictly operating-system specific, the sequence must include the following elements, or their equivalent:

1. The system stops issuing new requests to the add-in card driver or notifies the add-in card driver to stop accepting new requests.
2. The add-in card driver completes or terminates all outstanding requests.
3. The add-in card driver places the card in a state in which it will not initiate any interrupts or bus master activity on the bus

The operating system's implementation of quiescing add-in card activity must not depend on the selected add-in card ever coming back. In addition to quiescing add-in card activity, an operating-system vendor may optionally implement a less drastic "pause" capability, in anticipation of the same or a similar add-in card being reinserted. However, if the card is not reinserted, the old add-in card driver eventually must be quiesced.

Implementation Note: Quiescing an Add-in Card Driver

When the add-in card driver has been quiesced, it issues no bus transactions to the add-in card, even if another device sharing the same interrupt input as this driver instance generates an interrupt to its driver. In some operating systems, the add-in card driver is unloaded when it is quiesced.

An add-in card driver that controls multiple add-in cards must quiesce only the binding for the selected add-in card.

If an add-in card driver is being quiesced because the add-in card has failed, it may not be possible to complete some outstanding requests normally. Add-in card drivers should detect conditions when the add-in card does not respond properly, attempt to reset the add-in card, and terminate any outstanding requests.

4.1.3 Initializing the Configuration Space Header

Each operating-system vendor must specify the policy for initializing add-in card Configuration Space headers. The policy includes the following:

1. Whether initialization of the Configuration Space headers is the responsibility of Platform-specific software or operating-system-specific software.
2. If such initialization is the responsibility of Platform-specific software, then:
   a) Whether the operating system requires complete initialization of all resources or whether some resources are optional (such as doubly mapped I/O Space ranges)
   b) In what state a slot must be left if an error occurs during the initialization process.

The module responsible for initializing the Configuration Space header of a newly installed add-in card is defined to be the Platform Configuration Routine. It is responsible for assigning system resources, including I/O space, memory space, and PCI bus numbers, to all devices and functions on the new add-in card.

The operating-system vendor optionally specifies that configuration is an operating system function. In that case, the operating system is responsible for executing the Platform Configuration Routine after a slot is turned on.

Alternatively, the operating-system vendor optionally specifies that the Platform Configuration Routine must be handled by Platform-specific software such as the Hot-Plug System Driver. In that case, the operating system requires that the Hot-Plug System Driver execute the Platform Configuration Routine whenever it is given a request to turn a slot on.

In either case, the operating system must guarantee that the Configuration Space header is appropriately initialized before the add-in card driver is allowed to access the add-in card.

Implementation Note: Assigning Responsibility for the Platform Configuration Routine

The operating system determines which routine executes the Platform Configuration Routine based on how it tracks available configuration resources. For example, if the operating system depends upon the Platform BIOS to initialize the Configuration Space header at boot time, the Hot-Plug System Driver (supplied by the Platform vendor) might handle configuration when an add-in card is hot-plugged. However, if the operating system places the responsibility for configuring the header at boot time in higher-level routines, the Hot-Plug Service (supplied by the operating-system vendor) might handle configuration after a hot-plug event.

If the operating-system vendor requires the Platform Configuration Routine to be run by Platform-specific software, the operating-system vendor must also specify whether all resources must be assigned, or whether the operating system can accept a partial configuration, such as assigning memory resources but not I/O resources to an add-in card that requests both. The operating-system vendor must also specify whether the card is to be left in the on or off state if configuration fails.

An add-in card that has no Configuration Space header is treated as if all resource requests have been satisfied.

4.1.4 Add-in Card Option ROMs

Each operating-system vendor must specify the policy for the treatment of add-in card option ROM code after an add-in card is hot-inserted. Possible alternatives include, but are not limited to, the following:

1. Execution of an Open Firmware code image
2. Require that option ROM functionality be provided by some other means, for example, duplicating the function in the add-in card driver

Implementation Note: Replacing Functionality of Option ROMs on Add-in Cards

Some add-in card option ROMs establish essential operating parameters of the card. If these ROMs contain an image of Code Type 0 (Intel x86, PC-AT compatible), the code stored in this image is designed to execute during initial power-on, before the operating system is loaded, and, in general, cannot be executed during a hot-insertion operation.

If the operating-system vendor specifies that option ROMs are not executed after a hot-insertion, the add-in card vendor must use the capabilities and functions available through the operating system at run-time to replace the functionality provided by these option ROMs at boot-time. For example, the add-in card vendor may choose to implement this function by adding it to the add-in card driver itself, or by creating a separate card-specific configuration application that communicates with the add-in card driver.

4.1.5 Add-in Card Driver Requirements

An add-in card driver that supports removing and inserting an add-in card while the operating system is running has several requirements beyond those of a conventional add-in card driver.

First, it must be possible to quiesce and start the driver while the system is running. The operating-system vendor is responsible for specifying how the add-in card driver is to be quiesced and how it is notified to start using a new add-in card or to resume using an old one.

Second, the add-in card driver must guarantee that the card has completed its internal initialization sequence before the driver uses the card. (In some cases the add-in card driver becomes active much sooner after a hot-insertion event than it would after power is initially applied to the system.)

Third, one alternative for the treatment of PCI options ROMs is for the operating-system vendor to require that the add-in card driver replace any necessary functionality of option ROMs after a hot-insertion event (see Section 4.1.4).

4.1.6 Attention Indicator

The Platform is required to provide an Attention Indicator with each hot-plug slot (see Section 3.1.1). The Hot-Plug Service controls the state of this indicator by issuing Hot-Plug Primitives to the Hot-Plug System Driver.

This specification does not define the conditions under which the Hot-Plug Service must activate the Attention Indicator. The Hot-Plug Service is permitted to activate the

Attention Indicator at any time to call the user's attention to a particular slot. For example, the Hot-Plug Service might activate the Attention Indicator when the system detects problem conditions that require user intervention at the add-in card or the Hot-Plug Service might allow the user to specify functions such as locating all the 66-MHz slots or locating a particular slot so an associated cable can be changed.

4.1.7 Hot-Plug Service

The Hot-Plug Service is a broad collection of software routines, supplied by the operating-system vendor, that monitors and controls hot-plug operations, including the user interface and hot-plug sequence control. The Hot-Plug Service is responsible for issuing requests to the operating system to quiesce add-in card activity and to the Hot-Plug System Driver to turn slots on and off. It determines when it is appropriate for the system to stop or resume using an add-in card or to start using a new one.

The Hot-Plug Service must provide an interface to the user by which the user turns slots on and off. This interface is required to use the Physical Slot Identifiers provided by the Platform to designate slots. See Section 4.1.8 for more information.

The design and implementation of the Hot-Plug Service is unique to each particular operating system, and all other aspects of the Hot-Plug Service and its operation are controlled by the operating-system vendor.

4.1.8 Slot Identification

Several forms of slot identification are required throughout a hot-plug system. The first is the Physical Slot Identifier. The user interface in the Hot-Plug Service is required to identify slots by Physical Slot Identifiers provided by the Platform (see Section 4.1.7).

The second form of slot identification is PCI bus and device number. For example, the Hot-Plug System Driver is required to run PCI configuration cycles to read the card's 66 MHz Capable bit (see Section 4.2.2.3) and, in some systems, to run the Platform Configuration Routine (see Section 4.1.3). PCI configuration cycles require the add-in card in the slot to be identified by its PCI bus and device number.

The third form of slot identification is the Logical Slot Identifier. The operating-system vendor must specify the encoding of a parameter used by the Hot-Plug Primitives for uniquely identifying each slot. This parameter is called a Logical Slot Identifier. See Section 4.2.2 for a discussion of how the Hot-Plug Primitives use the Logical Slot Identifier.

The operating-system vendor must specify translation mechanisms between these three forms of slot identification, using information supplied by the Platform vendor (see Section 3.1.1).

Implementation Note: Examples of Translating Between Forms of Slot Identification

The Platform vendor supplies information about the way the Platform slots are wired in the BIOS ROM, the Hot-Plug Controller, Platform-specific code in the Hot-Plug System Driver, or any other suitable place. The operating-system vendor, in cooperation with the Platform vendor, defines the Logical Slot Identifiers and the translation mechanisms to use this information.

For example, in a single-chassis X86-class system that uses slot number for the Physical Slot Identifier, the BIOS ROM includes the translation between slot number and bus and device number in the IRQ Routing Table. The operating-system vendor might define the Logical Slot Identifier to be the physical slot number. In that case, the Hot-Plug System Driver would perform the remaining translation between Logical Slot Identifier and PCI bus and device number by reading the IRQ Routing Table.

Alternatively, the operating-system vendor might define the Logical Slot Identifier to be the PCI bus and device number. In that case, no translation is required by the Hot-Plug System Driver, but the Hot-Plug Service would access the IRQ Routing Tables to translate to physical slot numbers for the user interface.

In all cases, the translation mechanisms must comprehend the effect that adding and removing PCI-to-PCI bridges will have on the PCI bus numbers of other devices in the system.

4.2 Platform-Specific Software

4.2.1 Hot-Plug System Driver

The Hot-Plug System Driver is the device driver for the Hot-Plug Controller and is normally supplied by the Platform vendor. It is responsible for executing the Hot-Plug Primitive requests and providing results as described in Section 4.2.2.

A hot-plug system must include at least one Hot-Plug System Driver. Any given hot-plug slot is controlled by exactly one Hot-Plug System Driver.

4.2.2 Hot-Plug Primitives

The Hot-Plug Primitives presented in this section define what information is passed between the Hot-Plug Service and the Hot-Plug System Driver. Although this specification presents the Primitives in the form of requests and parameters, the actual programming interface is operating-system dependent and *not* controlled by this specification. Furthermore, the operating-system vendor is permitted to combine or split primitives into any number of specific operations.

Only requests, parameters, and error conditions specifically related to hot-plug operations are specified here. It is assumed that each operating system includes other procedural constraints and operating-system-specific error conditions, such as invalid-parameter errors and additional requests for diagnostic and system-management functions.

Although a system is permitted to have more than one Hot-Plug System Driver, this section assumes the Hot-Plug Service addresses the particular Hot-Plug System Driver that controls the slot of interest. Alternative implementations for addressing the proper Hot-Plug System Driver are allowed. For example, the Hot-Plug Service would be permitted to broadcast a Primitive to all Hot-Plug System Drivers and require the Hot-Plug System Driver for the slot of interest to execute the Primitive and all other Hot-Plug System Drivers to ignore it.

Unless otherwise specified, the parameters shown for each Hot-Plug Primitive assume that the minimum hot-plug Platform hardware described in Section 3.1.1 is implemented. If the Platform implements additional features, additional parameters may be needed for some or all Hot-Plug Primitives.

Implementation Note: Implementation Options for Hot-Plug Primitives

Although it is possible to implement the hot-plug primitives as single function calls or messages, this may not yield the most usable implementation. This technique yields an awkward interface where the Hot-Plug Service needs to pass down the current value of the parameter it doesn't intend to set when using the Setting Slot Status primitive. It also implies that the Hot-Plug System Driver needs to handle set-slot-on requests while on and set-slot-off while off.

Similarly, if gathering all the information in Querying Slot Status is undesirable in some way (because it is slow, causes power cycling of the card, and so on), it would be appropriate to implement Querying Slot Status as multiple requests so that the Hot-Plug Service asks for only the items of interest.

An operating system vendor may wish to split Setting Slot Status into a Set Attention-Indicator function and a Set Slot State function to relieve the Hot-Plug Service from providing the correct current value of the parameter it doesn't wish to modify. Alternatively, an operating system vendor may choose to have three values for each parameter: on (attention), off (normal), or no-change, and retain a single call or message per primitive. Similar considerations apply to the design of the calls or messages to implement the Querying Slot Status primitive, if the operating system designer believes there is some advantage in gathering only the parameters in which the Hot-Plug Service is interested.

In the following definitions of individual Hot-Plug Primitives, braces {} enclose a list of mutually exclusive alternative values for the parameters in question.

4.2.2.1 Querying the Hot-Plug System Driver

Parameters passed:

- none

Parameters returned:

- set of Logical Slot Identifiers for slots controlled by this Hot-Plug System Driver

The Hot-Plug System Driver must report the set of Logical Slot Identifiers for the slots it supports when requested by the Hot-Plug Service.

4.2.2.2 Setting Slot Status

Parameters passed:

- Logical Slot Identifier
- new slot state: {off, on}
- new Attention-Indicator state: {normal, attention}

Parameters returned:

- request completion status: {status change successful, fault—wrong mode/frequency, fault—not enough power available, fault—insufficient configuration resources,[17] fault—power failure, fault—general failure}

This request controls the state of the hot-plug slot and the state of the Attention Indicator for the slot.

The slot's state is either off or on. In the off state, the slot is powered down and isolated from the bus and it is safe to remove or insert an add-in card. In the on state, the slot is powered and, if an add-in card is present, it is ready for the Configuration Space header to be configured (or has already been configured, see Section 4.1.3). Although the hardware passes through multiple intermediate states between off and on, the Hot-Plug Service is only concerned about these two.

See Section 4.1.6 for a complete description of the Attention Indicator.

The Hot-Plug System Driver must enforce the requirement of T_{rhfa} specified in PCI 2.2. The Hot-Plug System Driver must wait at least T_{rhfa} after **RST#** has been deasserted to the slot before it accesses the card or completes a request to turn a slot on.

This request completes in a number of ways. If the slot is successfully turned on or off, the request returns the "successful" completion status. Unless otherwise specified, if a fault occurs while attempting to turn on a slot, the Hot-Plug System Driver must leave the slot in the off state, and the completion status indicates one of the following types of errors occurred. Some of these error conditions require optional Platform support (see Section 3.1.1). The Hot-Plug Service is required to support all of these error conditions. The Hot-Plug System Driver optionally supports a subset of the failure codes and reports the rest as "fault—general failure." However, it is recommended that the Hot-Plug System Driver implement codes for each failure type that the Platform detects.

- If the add-in card does not support the present mode or frequency of the bus, the completion status is "fault—wrong mode/frequency." As specified in PCI-X 1.0a, device mode and frequency capability is one or more of the following:

  Conventional PCI 33
  Conventional PCI 66
  PCI-X 66
  PCI-X 133

- If the Hot-Plug System Driver determines by reading the **PRSNT[1::2]#** pins on the add-in card that the Platform does not have enough power available to turn on the slot, the completion status is "fault—not enough power available."

[17] Applies only if the Platform Configuration Routine is executed from the Hot-Plug System Driver.

- If the Hot-Plug System Driver is responsible for running the Platform Configuration Routine, but the configuration failed because there were insufficient resources available, the completion status is "fault—insufficient configuration resources." The operating-system vendor must specify whether the slot is to be left on or off after this error. If the operating system accepts a partial configuration of an add-in card, the operating-system vendor must also specify how the Hot-Plug System Driver must indicate this condition. See Section 4.1.3 for more details.
- If a slot power fault is detected while turning on the slot, the completion status is "fault—power failure."
- If the Hot-Plug System Driver is not successful in turning on the slot (and configuring the card, if appropriate) for any other reason, the completion status is "fault—general failure."

4.2.2.3 Querying Slot Status

Parameters passed:

- Logical Slot Identifier

Parameters returned:

- Slot state {off, on}
- Add-in card power requirement {not present, low, medium, high}
- Add-in card mode/frequency capability
  Conventional PCI {33 MHz, 66 MHz, insufficient power},
  PCI-X {not capable of PCI-X mode, 66 MHz, 133 MHz}
- Slot mode/frequency {conventional PCI 33, conventional PCI 66, PCI-X 66, PCI-X 100, or PCI-X 133}

This request returns the state of the hot-plug slot and any add-in card that is present.

The add-in card power requirement parameter returns the information encoded by the add-in card on its **PRSNT[1::2]#** pins, independent of whether the slot is on or off. The exact encodings are specified in PCI 2.2.[18]

The add-in card mode/frequency capability indicates the maximum operating frequency of the device in conventional and PCI-X modes. The Hot-Plug System Driver accesses the **M66EN** and the **PCIXCAP** pins and/or the 66 MHz Capable bit, the PCI-X Capability Structure, and the 133 MHz Capable bit in the add-in card's Configuration Space to determine the card's mode and maximum frequency capability. This determination must be made independent of the present mode and speed of the bus, and independent of whether the Hot-Plug Service has turned the slot on or off. If the Hot-Plug System Driver cannot turn the slot on to read the **M66EN** pin or the 66 MHz Capability bit because the card power requirement exceeds the power budget, the Hot-Plug System Driver must indicate that there is insufficient power available to determine the add-in card's conventional PCI frequency capability. Add-in card frequency capability is meaningless if no add-in card is installed in the slot.

[18] PCI 2.2, Section 4.4.1.

The Slot mode/frequency parameter indicates at what mode and frequency the bus for this slot is currently operating, regardless of whether there is an add-in card in the slot or not, and whether the slot is on or off.

4.2.2.4 Asynchronous Notification of Slot Status Change

Parameters passed:

- Logical Slot Identifier

If the Hot-Plug System Driver detects an unsolicited change in the status of a slot (for example, a run-time power fault in a slot or a new card installed in a previously empty slot), it notifies the Hot-Plug Service using this Hot-Plug Primitive.

Asynchronous notifications are not required for standard hot-removal and hot-insertion operations, because these operations must follow orderly procedures defined by the operating-system and Platform vendors. The primary use of asynchronous notifications is to keep the Hot-Plug Service informed of changes to the state of the system outside of such operations.

It is recommended that the Hot-Plug Service not use asynchronous events to automatically change slot state, because, in some cases, an end-user inserts an add-in card and immediately removes it or repeatedly removes and inserts it until it is properly seated.

Appendix A
Presence and Capability Signals

Signals used to communicate the presence of an add-in card and its capability are **PRSNT[1::2]#, M66EN**, and **PCIXCAP**.

PRSNT[1::2]# are pulled up to main power (on the system side of the slot-specific power switches) so that the Hot Plug Controller can detect the presence and power requirements of a card before power is applied to the slot.

The **PCIXCAP** signal is trinary (3-level) in nature, as defined by PCI-X 1.0a. The means of sensing the three levels of this signal is system-dependent. (See PCI-X 1.0a for implementation options.)

The **M66EN** pin is pulled up to (slot-specific) switched power. This pin is defined in PCI 2.2 to be a card input (as well as output)[19] and, therefore, the Platform is unable to source current into this pin unless the card is powered. After power has been applied to the slot and the Hot-Plug Controller has sampled the capability of the add-in card, the Hot-Plug Controller is responsible for driving the **M66EN** input low if the bus is operating in conventional PCI 33 MHz mode. The Platform must ensure that all cards on the same segment see a consistent value on their **M66EN** inputs during the rising edge of the slot specific **RST#** signals.

[19] PCI 2.2, Section 7.5.1.